The Protein Threading Problem is in P?

Nicola Yanev, Rumen Andonov

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

The Protein Threading Problem is in P?

Nicola Yanev — Rumen Andonov

N° 4577

October 2002

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--4577--FR+ENG

The Protein Threading Problem is in P?

Nicola Yanev* , Rumen Andonov†

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet SYMBIOSE

Rapport de recherche n° 4577 — October 2002 — 15 pages

Abstract: This work is about a problem from computational biology known as *protein threading problem.* By finding out an appropriate linear mixed-integer programming (MIP) formulation we demonstrate that the real-live instances of this problem could be efficiently solved by using only some linear-programming (LP) solver instead of special-purpose branch&bound algorithm. This is due to the fact that within the frame of MIP model proposed, all biological instances, we were able to test, attain their optima at feasible vertices of the underlying LP polytope which is the essence of the statement in the title.

Key-words: protein threading problem, mixed-integer programming

The work of N. Yanev has been partially supported by the GenoGRID project (ACI GRID, Ministère de la Recherche) and was performed during a visit to the SYMBIOSE project, IRISA, Rennes

* University of Sofia, Bulgaria, choby@math.bas.bg
† INRIA, Rennes, France, randonov@irisa.fr. On leave from LAMIH/ROI, University of Valenciennes, France, Rumen.Andonov@univ-valenciennes.fr

Unité de recherche INRIA Rennes

Le problème de reconnaissance de repliement de protéines est-t-il dans P?

Résumé : Cet article concerne le problème de reconnaissance de repliement de protéines connu sous le nom de "*protein threading problem*". Nous proposons plusieurs formulations MIP (*Mixed-Integer Programming*) du problème et nous montrons que toutes les instances basées sur des données "réelles" (utilisées par les biologistes) peuvent être résolues par un logiciel LP (*Linear Programming*) au lieu d'un algorithme b&b (*branch&bound*) dédié. Dans le cadre de notre dernier modèle MIP, l'optimum de tous les cas que nous avons résolus, est atteint dans un sommet (0,1) du polytope sous-jacent, ce qui explique le titre de cet article.

Mots-clés : protein threading problem, mixed-integer programming

1 Introduction

Although, biological results are beyond the scope of this paper and for the reader, interested in "why the protein threading problem ?" question it is better to see [1, 5, 8, 9, 10], we will try to describe the reasons behind briefly. The protein folding problem also referred as *the holy grail of molecular biology* or *the second half of the genetic code* is to determine the three-dimensional folded shape (protein structure prediction) of a protein (sequence of characters drawn from an alphabet of 20 letters). It is important because the biological function of proteins underlies all life, their function is determined by their three-dimensional shape, and their shape is determined by their one-dimensional sequence. The importance of computational solutions increases due to the explosion of sequences becoming available, compare to the slow growth in the number of experimentally determined three-dimensional structures. The direct approach to protein folding is based on modeled atomic force fields and approximations from classical mechanics and still faces stiff challenges for large proteins. Another approach is to exploit the fact that the amino acid types have different preferences (expressed by a function f (say)) for occupying structural environments (*alpha-helices* or *beta-sheets,* so called segments here below). Additionally, some of the approaches exploit the fact that there appear to be distinct preferences for spatial proximity as a function of those environments (the *local* and *non-local* interactions and functions p_{ij} scoring the pairwise interactions between the i^{th} and j^{th} segments). These interaction preferences have been quantified statistically and used to produce a score function reflecting the extent to which the amino acids from the sequence are located in preferred environments and adjacent to preferred neighbors. This done, a target sequence(query) can be aligned or threaded into a template structure(core) by searching for threading which optimizes the score function.

More formal presentation of the problem will be done by simultaneously introducing of an already existing terminology. Let C, called *core* be a set of m items S_i, called *segments*, of length l_i. This set must be *aligned* to a sequence L of N characters from some finite alphabet. Let t_i be the position in L where S_i starts. An alignment is called *feasible threading* if:

i. $t_i \geq t_{i-1} + l_{i-1}$ for all i;

ii. the length g_i (called *gap* or *loop*) of uncovered characters, i. e. $g_i = t_i - t_{i-1} - l_{i-1}$ is bounded, say $g_i^{min} \leq g_i \leq g_i^{max}$.

Each feasible threading $t = (t_1, t_2, ..t_m)$ is scored by a function $f(t) = \sum f_i(t_i) + \sum h_i(g_i)$, where f_i scores the placement of a segment i to a given position t_i and h_i is used in some experiments for scoring the gap between two consecutive segments. If the problem now is to minimize $f(t)$ over the set F of feasible threadings, one can show (see section 3) the equivalence with the shortest path problem between two vertices of a very structured digraph. What makes the problem interesting (and intractable–NP-hardness is proven in [3]) is the the paths lengths increase by additional costs in the following way: for some given subset S' of the set of all pairs (unordered) of segments, the length of the path $(t_1, t_2, ..., t_m)$ is increased by $\sum p_{ij}(t_i, t_j)$ where the sum is taken over all $(i, j) \in S'$. The function p_{ij} scores the pairwise interaction between segments S_i, S_j in dependence with their

alignment t_i, t_j with L. One can also show (again in section 3) that if for all $(i,j) \in S'$, $j = i+1$ (*local interactions*) then again we obtain an equivalent (easy solvable) shortest path problem. So what makes the problem difficult is the presence of *non-local interactions*, i. e. $j > i+1$.

This mathematical model is the one we started with and what is still considered as a challenge is finding of a really fast algorithm for solving the abovementioned optimization problems. The main inspiration for us to start working over this subject are the results announced in [1, 2] where a branch-and-bound algorithm (b&b) is described which is very successful over a big set of real-life (biological) examples. We were aware of the fact that the real-life instances are may be more tractable than randomly generated instances, but even then, a branch-and-bound algorithm able to solve non-linear integer programming problem over the search space of size up to 10^{31} feasible threadings by using a relaxation of non-evident quality could become a reason for asking questions like: i) how much hard is the problem on biological instances (especially for so called self threading)?; ii) could it be improved (either by improving bounds or by creating another mathematical programming model)?; iii) is it necessary to write special purpose code or some existing solvers could be used? The answers of these questions is given in the subsequent sections.

2 Linear mixed-integer programming formulations

The first and most obvious model for the protein threading problem from the introduction is to minimize the objective function

$$\sum_i \sum_j g_{ij} x_{ij} + \sum_{(i,k)\in E} \sum_{j \le l} \bar{g}_{ijkl} x_{ij} x_{kl}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, m \tag{1}$$

$$x_{ij} \le \sum_{k=1}^{j} x_{i-1,k}, \qquad i = 2, m; j = 1, n-1 \tag{2}$$

where m is the number of segments, $n = N - \sum_{k=1}^{m} l_k + 1$(the numbers l_k are the lengths of the segments increased by l_k^{min}- the minimal number of gaps between the segments k and $k+1$) is the number of possible placements of each segment relative to the end of the previous one, x_{ij} are binary variables with $x_{ij} = 1$ meaning the segment i starts from the absolute position $\sum_{k=1}^{i-1} l_k + j$ of the protein sequence L. The meaning of (1) is obvious and (2) cares for non-overlapping placement of segments. The nonlinear term in the objective function is for the local and non-local interactions between the segments, i. e. it is supposed that a nonempty set $E = \{(i,k), k > i\}$ of segments pairs (i,k) is given together with scores $\bar{g}_{ijkl}$ for placement S_i at the j^{th} relative position and S_k at l^{th} one.

In order to linearize the problem we introduce new variables $z_{ijkl} \in [0,1]$ in the objective function instead of $x_{ij}x_{kl}$ and add the constraints

$$z_{ijkl} \leq x_{ij}, \;\; z_{ijkl} \leq x_{kl}, \quad \forall (i,k) \in E; \; l \geq j \tag{3}$$

$$x_{ij} + x_{kl} - z_{ijkl} \leq 1 \tag{4}$$

In this way we obtain a linear mixed integer programming problem of minimizing

$$\sum_i \sum_j g_{ij} x_{ij} + \sum_{(i,k) \in E} \sum_{j \leq l} \bar{g}_{ijkl} z_{ijkl}$$

subject to (1) (2) (3) (4) and $x \in \{0,1\}$, $0 \leq z \leq 1$. We will denote this model by **M1**.

For the further discussions it is worthwhile to list some easily derivable observations:

Obs. 1: The number of feasible x is given by $N_x = \binom{n-1+m}{m}$ and could activate at most $N_z = \binom{n-1+k}{k}$ pairwise interactions. Here k is the number of segments, covered by all local and non-local interactions. So when $k < m$, the models based on z to be integer instead of x will have smaller search space. (Further we will show how to reduce k to be the number of segments covered by the non-local interactions only).

Obs. 2: Adding a constant to all coefficients of the objective function does not change the set of optimal solutions.

Obs. 2 allows for considering instances with all objective function coefficients negative (in the biological examples they are of different signs), which in turn allows to discard all the constraints (4). However, this option was never used because of the weakness of the LP-bounds, used for fathoming the nodes of branch&bound tree. It is demonstrated in [4] that (3) and (4) give the tightest (in the sense of underlying polytope) possible linearization of the logical implication $z = 1$ iff $x = 1$ and $y = 1$. It is important to say now, that this and all subsequent MIP (from Mixed-Integer Programming) models were intended to be solved by CPLEX solver of ILOG version 7. 1, and not by some special-purpose solver (if any?). Unfortunately, this model, mainly because of the weakness of LP-bounds (more detailed reasoning is given in table 1), was never efficiently solved by CPLEX and we leave it with some lessons learned.

3 Network flow formulation

Let $G(V,E)$ be a digraph with arc set $E = \{((i,k),(i+1,l)) \mid i = 1,m-1; \; l \geq k\}$ and vertex set $V = \{(i,k) \mid i = 1,m; \; k = 1,n\}$, and let $NL = \{(i_1,j_1),...,(i_t,j_t)\}$ be the set of non-local $(i_s < j_s - 1)$ interactions. This induces the set of indices $E^{ind} = \{(i_k l j_k f) \mid f \geq l, k = 1,t; \; l = 1,n\}$ meaning that the left segment i_k of the non-local pair $(i_{k,}, j_k)$ is placed at position l and the right segment j_k of the same pair at position f. The condition $f \geq l$ is in accordance with the sub-paths

in G from the vertices in layer i_k to the vertices in layer j_k. We introduce also variables x_e for $e \in E$ and z_e for $e \in E^{ind}$. Now we can model the problem from section 2 in the following way:

$$\sum_{e \in E} c_e x_e + \sum_{e \in E^{ind}} c_e z_e \Rightarrow \min \tag{5}$$

such that:

$$\sum_{e \in \Gamma(i,k)} x_e - \sum_{e \in \Gamma^{-1}(i,k)} x_e = 0 \quad \forall (i,k) \in V \tag{6}$$

$$\sum_{e \in \Gamma(S)} x_e = 1 \tag{7}$$

$$z_c \leq \sum_{e \in \Gamma(i_k,l)} x_e, \quad z_c \leq \sum_{e \in \Gamma^{-1}(j_k,f)} x_e \quad \forall c = (i_k, l, j_k, f) \in E^{ind} \tag{8}$$

$$\sum_{e \in \Gamma(i_k,l)} x_{e+} \sum_{e \in \Gamma^{-1}(j_k,f)} x_e - z_c \leq 0 \quad \forall c \in E^{ind} \tag{9}$$

$$\sum_{f \geq l} z_{i_k l j_k f} = 1 \quad \forall (i_k, j_k) \in NL \tag{10}$$

$$z - \text{ binary}, x \geq 0 \tag{11}$$

Into this model S is an artificial source node (see fig.1), $\Gamma(x)$ is the set of arcs outgoing from vertex x and $\Gamma^{-1}(x)$ is the set of in-going arcs. Constraints (6), (7) are network flow representation of the paths from S to T (see fig.1), constraints (8) force the path to pass through the pair of vertices activated by SOS (Special Ordered Set) constraints (10), constraints (9) are for tightening the LP-relaxation. This model was built under the presumptions of Obs.1, i.e. to enumerate the smaller search space of variables z than of x, that will be the case if we switch the integer requirements from z to x variables (this need some minor changes in the model). From the well known properties of the network flow polytope one can see that for each fixation of z variables to $0, 1$ the respective vertices of the underlying polytope in $(x, z)-$space have $(0, 1)$ x coordinates. To conclude we need to say how the arcs weights c_e are related to the scores from the introduction. Each weight is a sum of three numbers: one for the tail of the arc (segment-to-position cost), second for the gap cost between segments(if any) and the third is for the local interactions (if any). If the leading and/or trailing gaps are scored then these numbers are prescribed to the out-going arcs from S and/or in-going arcs in T. From the graph in fig.1 we could have more geometrical insight for the problem of optimal aligning of some sequence with a core of 5 segments, each one with three possible placements: the path given in a thick lines has a length 5 but taking into account the pairwise interactions (in this case (1,1,3,2), (3,2,5,2) - the path passes through the vertices (1,1),(3,2), (5,2)) we must add the

costs for passing through these vertices ($c_{1132} + c_{3252}$) to 5 and to obtain the actual length 14 of the path (threading). Thus if we have given weights to all arcs and a table of the costs for the designated non-local pairwise interactions the optimization problem will convert to finding a path from S to T with minimal updated length. From this figure we can also stress z- to- x relation: once the z variables are fixed ($z_{1132} = 1, z_{3252} = 1$, all other are zero because of (10)) to find which x will be fixed to 1 is equivalent to find the shortest from among the paths passing through (1,1), (3,2), (5,2). Within this framework we can answer the question for the effectiveness of an existing branch&bound algorithm.

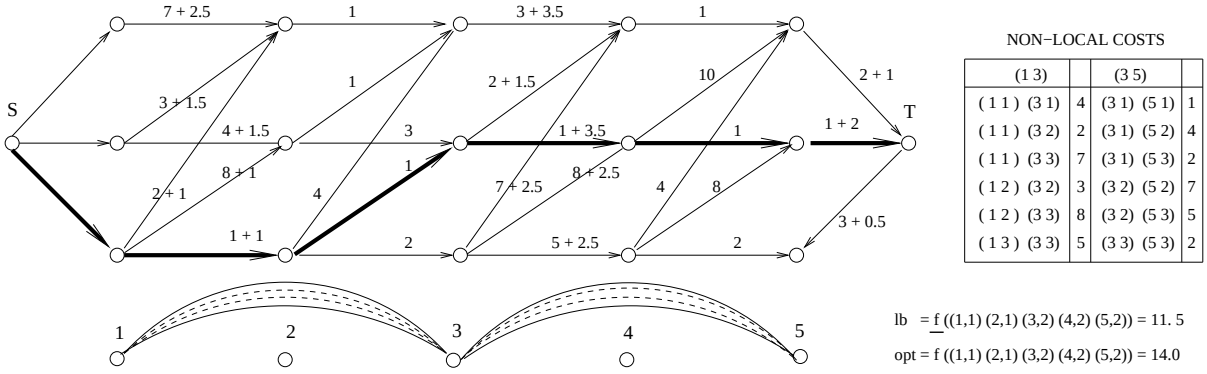

Figure 1: Graph corresponding to the network flow formulation of the problem. $f(x)$ is a function computing the exact cost of the path x, while $\underline{f}(x)$ computes its lower bound according to [1].

4 Related works

In [1, 2] a branch&bound algorithm (later referred to as **LS**) is given together with the demonstration of its effectiveness on a wide range of instances. The idea of the algorithm is simple and could be now easily explained. The idea (in the framework of our graph setting) of the lower bound is based on the following: let $(1, i_1), (2, i_2), ...(m, i_m)$ be an arbitrary path in the graph G and let (k, l) be a vertex on this path s.t. $\{(s_1, k), ...(s_p, k), (k, r_1), ..., (k, r_c)\} \in NL$ (k is left or right end of at least one non-local relation). Let $S_L = \{(s_1, k), ...(s_p, k)\}$ and $S_R = \{(k, r_1), ...(k, r_c)\}$. Then the number

$$l^*(k, l) = 0.5\min_{i \le l} \sum_{s_i \in S_L} c_{s_i ikl} + 0.5\min_{l \le r} \sum_{r_j \in S_R} c_{klr_j r}$$

is a lower bound to the costs incurred by the non-local relations having (k, l) at one of their ends [1, 2]. Now we can add $l^*(k, l)$ (shown on fig.1 as the second term in the sums over some

arcs) to the lengths of all arcs out-going from (k,l). Having done this for all vertices covered by the non-local relations then the shortest path from S to T in the so updated graph G is a lower bound for the objective function (5). The $O(m^2n^2)$ complexity of this procedure [1, 2] could be derived from the complexity of the shortest path problem and the complexity of the algorithm for l^* computation (on fig.1 `lb` is the bound obtained by this relaxation and `opt` the optimal value). In the same work a list of impressive computational results is given on a reach set of s.c. self-threading (the protein sequence is aligned with its own core) instances. When we run CPLEX on the MIP models generated according to (5)-(11) on a large subset of these instances the results are always: *the LP relaxation attains its minimum at a feasible (0,1) vertex, hence optimal.* This property is so pertinent to the model that one could use it for defining the self-threading subclass. The relaxed problem used in **LS** model [1, 2] is based on minimizing of a function $\underline{f}(x)$ which is inferior to the objective function $f(x)$ over the set of feasible threadings. For such a relaxation, an optimal solution x^* to the relaxed problem is optimal for the original one if $f(x^*) = \underline{f}(x^*)$. For the **LS** model this could be taken as the self-threading subclass defining property and this is the reason for the effectiveness of the **LS** algorithm on the instances of this subclass. What is important to add here is that all these properties are score dependent and they could be lost once the scoring scheme changed. For all instances reported below the objective function coefficients are generated by using FROST (*Fold Recognition Oriented Search Tool*) software [6, 7]. It is beyond our knowledge to judge on diversities or common features of the scoring schemes used in different biological laboratories. At least for the FROST scheme, the following example could shed some light on: for the `1gal_0` sequence aligned with `1ad3a0` core (these are standard PDB (Protein Data Bank) notations), 953 174 from among 1835394 segment interactions coefficients are distinct–and all this diversity is a result from simple composition of functions defined over 20 letters alphabet[1].

5 Further improvements of the MIP model

In order to improve the LP-bounds and the CPLEX branching strategy by imposing branching on the SOS constraints instead of on a single variable (but at the expense of adding extra constraints) the following modification of the model (5)-(10) is done:

Let $L(NL) = \{i_s | (i_s, j_s) \in NL\}$, $R(NL) = \{i_s | (j_s, i_s) \in NL\ \}$.

$$y_{il} = \sum_{f \geq l} z_{iljf} \qquad l = 1, n; \ \ \forall i \in L(NL) \ \ (i,j) \in NL \tag{12}$$

$$y_{il} = \sum_{f \leq l} z_{jfil} \qquad l = 1, n; \ \ \forall i \in R(NL) \ \ (j,i) \in NL \tag{13}$$

$$\sum_{l=1}^{n} y_{il} = 1 \quad \forall i \in R(NL) \bigcup L(NL) \tag{14}$$

[1] The data variety relates the combinatorial hardness of the optimization problem.

$y-$binary and all z variables included into these definitions change from binary to continuous. The role of the newly introduced constraints is twofold: i) (12)-(14) are tighter than (10) thus improving LP-bounds and could be used for much more flexible branching strategy through explicitly introducing them in the SOS section of the model to be solved by CPLEX ; ii) the number of the (0,1) variables is drastically reduced because the z variables, covered by the defining constraints, are forced to be integer in consequence of the unimodularity of the corresponding matrix. Although, computationally this model (denoted here by **M2**) is a significant improvement over the preceding ones, namely its was successfully run (by CPLEX) on a set of biological examples, its main drawback is the size of the problems created (see table 1). For instance, when aligning the protein `1coy_0` with the `core 1gal_0` (a problem of size 36 segments and 81 positions) we observe that the corresponding MIP problem has 741 264 rows 360 945 columns and 54 145 231 nonzero elements. This is, of course, prohibitively large for practical use and appeal for splitting techniques (see section 7) which help for significant reduction of the MIP problem size and also of the solution time. One could feel that what impacts mainly on the size of the optimization problem (besides m, n) is the two level control adopted: x variables are controlled by z variables which are controlled by y variables. How to overcome this deficiency of the model is described in the next section.

6 The main result

Now we introduce binary variables y_{ij} , $i \in L(NL) \cup R(NL)$ $j = 1, ..., n$ to prevent the flow passing through vertex (i, j) when $y_{ij} = 0$ or direct through it when $y_{ij} = 1$. This could be modeled in an obvious way to obtain the following network-flow alike problem (**M***):

$$\sum_{i=1}^{m-1} c_i x_i + \sum_{i \in NL} c_i z_i \Rightarrow \min \tag{15}$$

subject to

$$A_i z_i - I y_i = 0 \quad i \in L(NL) \cup R(NL) \tag{16}$$

$$B_i x_i - I y_i = 0 \quad i \in L(NL) \cup R(NL) \tag{17}$$

$$Fx = l \tag{18}$$

$$Ey = 1 \tag{19}$$

$$y_{ij} \in \{0, 1\} \tag{20}$$

where A_i are node-arc incidence matrix for the nodes $(i,1),..(i,n)$ and the z-arcs out-going/in-going from/to these nodes, B_i are node-arc incidence matrices for the same nodes but for the $x-$arcs, F is the node -arc incidence matrix for the digraph G, thus (18) are the classical network flow constraints, l is zero vector except for the first n components equal to y_{1j} and finally (19) which are SOS constraints on (0,1) variables y. All matrices are absolutely unimodular and the problem is of block angular form with y as the only binding variables.

Starting tests with this model on various biological examples, surprisingly the *LP solution happened to be integer* for very large set of instances. Thus, as a side effect in looking for efficient algorithm for protein threading problem we obtain the main result, stated as the following :

CONJECTURE: *The protein threading problem is in P.*

More precisely, the claim is as follows: each one of the models LS, M1, M2, or M* could be considered as a formal definition of protein threading problem. Now, if the set of query-to-core instances is restricted to real ones and the score function is as in [6, 7] then an algorithm exists which can solve each such problem instance in polynomial time. A possible relaxation of the score function constraint could be easily checked by a simple change in the generator of the objective function coefficient in the model M*. The problem is, which scoring schemes are welcomed by the biological society. One could go further with the enlargement of the score function class, but it is better to stop here.

What is seen from the results in tables 1 and 3 is that there is enough evidence to believe this conjecture is true. Even for polytope of at least 10^{39}vertices, the objective function of the relaxed problem of (15)-(20), counting more than million variables, attains its minimum at a vertex of (0,1) coordinates. It seems that the nature of the scores adopted into the protein threading definition allows for discarding of a lot of non-local interactions without affecting the optimal point. If this is true, then the claim of the conjecture will follow from the existence of a polynomial algorithm for the problem with a special non-local interactions structure. Whatever is the case, it means that only by using the M* model and the non-special purpose LP solver of CPLEX one could solve in affordable time all practical problems in the context of protein threading. Some hints for further improvement of the effectiveness of the approach is discussed in section 7. As for the minimalilty of polytope describing constraints, we must note that each attempt of aggregation, say in (16), spoils the feasibility of the LP solution.

7 Split and conquer

Let us recall here that by the very definition of (15)-(19) model, one can partition (split) it to smaller subproblems by imposing constraints on the paths in the graph G in the following way. Let k be some segment and (k,j) $j = 1,...n$ are the vertices in the k^{th} layer. Then by partitioning the set $\{1,2,...,n\}$ into r intervals, we could split the problem into r subproblems of smaller sizes and the i^{th} one is defined over the paths in G which pass through the vertices in the i^{th} interval only. Thus

if this interval is (p, q) then for the segments l on the left(right) of k only the vertices $(l, s) : s \leq q$ ($(l, s) : s \geq p$) should be taken into account. The best choice for k is $\lceil 0.5m \rceil$ and if the goal is to split the problem into subproblems of approximately equal size then the intervals should be of equal length. One could target splitting on the criterium - almost equal number of paths, but because of simplicity of the implementation we used the first approach. Now, if the split is done one can start solving the subproblems in some order by passing the best objective function value found as a cutoff for the subsequent subproblems. Thus, by having the chance to start with the subproblem which contains the optimal path, all other subproblems will be aborted by the LP solver at the moment when dual objective reaches the cutoff value. The effect of this strategy is demonstrated in table 3 on instances which are worthwhile to split. One good choice for the problems to start with is based on the observation that for the biological instances the optimal path passes "near" the middle vertex in the middle layer (this area is a crossroad of maximal number of paths). The estimate of this nearness could be given as a parameter for sequential or as well for parallel implementation of such splitting approach.

model	query name	core name	size		space size	LP size		iter	time
			segm.	pos.		rows	columns		
LS	1bdo_0	1pauB0	8	14	2.03e+05	-	-	-	0.2s
M1	1bdo_0	1pauB0	8	14	2.03e+05	1795	1792	22873	22.4s
M2	1bdo_0	1pauB0	8	14	2.03e+05	3505	1864	3072	29.3s
M*	1bdo_0	1pauB0	8	14	2.03e+05	439	1853	283	0.2s
LS	1cydA0	1eny_0	16	40	8.14e+06	-	-	-	2m 19s
M1	1cydA0	1eny_0	16	40	8.14e+06	28515	28520	1181829	>12214s
M2	1cydA0	1eny_0	16	40	8.14e+06	58962	31673	5072	2m 35s
M*	1cydA0	1eny_0	16	40	8.14e+06	2909	31654	1676	4s

Table 1: Small instances: four models comparisons

8 Computational experiments

In the tables 2, 3 we summarize the results from running CPLEX on SUNW, UltraSPARC-II, 400 MHz, CPU computer. The instances (scores) are drawn from a redirected output of FROST [6, 7] which tries to find the best fit of multiple queries-to-multiple cores bank. The LS algorithm [2] is used at a final stage of this complex and time-consuming procedure. In order to generate interesting (real big) instances we have to limit the time for this b&b code to some upper bound varying between 30min. and 2h. according the instance size. When LS reaches this bound (respectively indicated by the sign $>$ in the last "time" column) we write-down the best value ever found by b&b. This should be taken into account for some of the results in table 2 where we sacrificed the quality of

the comparison for the chance to test on instances, never attempted before. What we mean by interesting instance is one with more than 10^{31} feasible threadings.

9 Conclusions

We have demonstrated, once more, that the achievements of the mathematical programming theory and algorithms are valuable tools for attacking optimization problems this time arising in the computational biology. We succeed, relying on such achievements, to linearly model a problem of nonlinear combinatorial nature and to solve efficiently a lot of instances without having written a single line of some code (different from model builder). This model reveals an unexpected property of protein threading problem when it is considered only over biological instances, namely feasibility of the linear programming solutions. This will allow (at least up to the moment of finding counter-example) to improve the performance of the approach by looking for a LP solver oriented to the network flow structure of the LP model proposed. Finally, we believe that the conjecture in the title of this work could be much easily resolved by the combine efforts of biologists and operations research specialists.

model	query name	core name	problem size		space size	score	time
			segm.	pos.			
LS	2cyp_0	2cyp_0	15	98	1.5e+18	-1898.2	1m 45s
M*	2cyp_0	2cyp_0	15	98	1.5e+18	-1898.2	18s
LS	3grs_0	3grs_0	30	114	6.5e+30	-3809.7	7m 26s
M*	3grs_0	3grs_0	30	114	6.5e+30	-3809.7	1m 08s
LS	1coy_0	1coy_0	27	149	4.0e+31	-3386.2	20m 47s
M*	1coy_0	1coy_0	27	149	4.0e+31	-3386.2	1m 48s
LS	2cyp_0	1theA0	13	138	1.8e+18	-11.4	>20m
M*	2cyp_0	1theA0	13	138	1.8e+18	-11.6	10m 06s
LS	3minA0	4kbpA0	23	189	3.2e+30	57.42	>30m
M*	3minA0	4kbpA0	23	189	3.2e+30	57.42	53m 31s
LS	1coy_0	1gal_0	36	81	1.3e+30	100.0	>30m
M*	1coy_0	1gal_0	36	81	1.3e+30	98.7	7m 40s
LS	3minB0	1gpl_0	23	215	5.3e+31	120.4	>50m
M*	3minB0	1gpl_0	23	215	5.3e+31	63.5	46m 34s
LS	1gal_0	1ad3A0	31	212	1.3e+39	140.2	>1h 15m
M*	1gal_0	1ad3A0	31	212	1.3e+39	76.3	7h 10m 15s

Table 2: Huge instances: LS versus M* comparison. Note that even for selfthreading (the first three instances) M* model is much faster than LS which in this case ends after a single lower bound computation

query name	core name	problem size		space size	LP size		number of sub-problems	time
		segm.	pos.		rows	columns		
2cyp_0	1theA0	13	138	1.8e+18	9613	337419	1	10m 06s
2cyp_0	1theA0	13	138	1.8e+18	5764	163908	5	4m 00s
2cyp_0	1theA0	13	138	1.8e+18	6520	198225	3	3m 38s
2bmhA0	1cem_0	21	203	1.5e+29	23670	1225613	1	34m 45s
2bmhA0	1cem_0	21	203	1.5e+29	15398	620424	3	21m 12s
2bmhA0	1cem_0	21	203	1.5e+29	13643	491499	5	17m 54s
3minB0	1gpl_0	23	215	5.3e+31	24011	1305173	1	46m 34s
3minB0	1gpl_0	23	215	5.3e+31	12760	426889	10	29m 30s
3minB0	1gpl_0	23	215	5.3e+31	14045	507386	5	22m 03s
2cyp_0	3grs_0	30	219	4.1e+38	41472	2294957	1	58m 29s
2cyp_0	3grs_0	30	219	4.1e+38	22417	847060	9	49m 24s
2cyp_0	3grs_0	30	219	4.1e+38	24222	979800	5	32m 43s
1gal_0	1ad3A0	31	212	1.3e+39	37191	1993288	1	7h 10m 15s
1gal_0	1ad3A0	31	212	1.3e+39	14666	675237	13	3h 10m 53s
1gal_0	1ad3A0	31	212	1.3e+39	20296	718448	9	1h 14m 23s

Table 3: Huge instances: impact of split and conquer strategy in M* model

10 Acknowledgement

We are grateful to Jean-François Gibrat and Antoine Marin for introducing us the problem, for many helpful discussions and for providing us with the code of Lathrop&Smith algorithm as well as all data concerning the protein structure prediction problem. Special thanks are due to Stefan Balev who participated actively in the initial stage of this study.

References

[1] R. Lathrop, R. Rogers Jr., J. Bienkowska, B. Bryant, L. Butorovic, C. Gaitatzes, R. Nambudripad, J. White, T. Smith, Analysis and Algorithms for Protein Sequence-Structure Alignment, Comp Methods in Molecular Biology, chapter 12, pp. 227-283, 1998.

[2] R. Lathrop, T. Smith, Global Optimum Threading with Gapped Alignment and Empirical pair Score functions, J. Mol. Biol., 255, 641-665, 1996.

[3] R. Lathrop, The protein threading problem with sequence amino acid interaction preferences is NP-complete, Protein Eng. 7, 1059-1068, 1994.

[4] F. Plastria, Formulating logical implications in combinatorial optimization, EJOR 140, 338-353, 2002.

[5] Xu Y., Xu D., Uberbacher E., An Efficient Computational Method for Globally Optimal Threading, J. Comp. Biol., V5, Number 3, 1998.

[6] A. Marin, J.Pothier, K. Zimmermann, J-F. Gibrat, Protein structure prediction: bioinfromatic approach, I. Tsigelny Ed. International University Line, 2002, chapter "Protein threading statistics: an attempt to assess the significance of a fold assignment to a sequence"

[7] A. Marin, J.Pothier, K. Zimmermann, J-F. Gibrat, FROST: A Filter Based Recognition Method, to appear in Proteins: Struct. Funct. Genet., vol. 49, 2002

[8] J.C. Setubal, J. Meidanis, Introduction to computational molecular biology, 1997, International Thomson Publishing Inc.

[9] T. Lengauer, Computational Biology at the Beginning of the Post-genomic Era, LNCS, Vol. 2000, "Informatics: 10 Years Back - 10 Years Ahead", R. Wilhelm(Ed.), Springler, Berlin 2000, P. 341-355

[10] R. Thiele, R. Zimmer, T. Lengauer, Protein Threading by Recursive Dynamic Programming, J. Mol. Biol., 290, 3(1999), 757-779

Contents

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399